組合せ最適化入門：線形計画から整数計画まで

梅谷　俊治[†]

線形計画問題において変数が整数値を取る制約を持つ整数計画問題は，産業や学術の幅広い分野における現実問題を定式化できる汎用的な最適化問題の１つであり，最近では分枝限定法に様々なアイデアを盛り込んだ高性能な整数計画ソルバーがいくつか公開されている．しかし，整数計画問題では線形式のみを用いて現実問題を記述する必要があるため，数理最適化の専門家ではない利用者にとって現実問題を整数計画問題に定式化することは決して容易な作業ではない．本論文では，数理最適化の専門家ではない利用者が現実問題の解決に取り組む際に必要となる整数計画ソルバーの基本的な利用法と定式化の技法を解説する．

キーワード：組合せ最適化，線形計画，整数計画，数理最適化

Introduction to Combinatorial Optimization: Model Building in Integer Programming

Shunji Umetani[†]

The integer programming (IP) model is a general-purpose optimization model that can formulate a surprisingly wide class of real applications using integer variables in linear programming (LP) models. Recent development in IP software systems has significantly improved our ability to solve large-scale instances. However, it is still difficult for most non-expert users to formulate real applications into IP models, because all conditions need to be written in linear inequalities. This paper demonstrates how to use IP software systems and formulate real applications into IP models.

Key Words: *Combinatorial optimization, Linear Programming, Integer Programming, Mathematical optimization*

1　はじめに

線形計画問題において全てもしくは一部の変数が整数値を取る制約を持つ（混合）整数計画問題は，産業や学術の幅広い分野における現実問題を定式化できる汎用的な最適化問題の１つである．近年，整数計画ソルバー（整数計画問題を解くソフトウェア）の進歩は著しく，現在では数千変数から数万変数におよぶ実務上の最適化問題が次々と解決されている．また，商用・非商用を含めて多数の整数計画ソルバーが公開されており，整数計画問題を解くアルゴリズムを知らなくても定式化さえできれば整数計画ソルバーを利用できるようになったため，数理最

[†] 大阪大学大学院情報科学研究科, Graduate School of Information Science and Technology, Osaka University

適化以外の分野においても整数計画ソルバーを利用した研究が急速に普及している．

最適化問題は，与えられた制約条件の下で目的関数 $f(\boldsymbol{x})$ の値を最小にする解 $\boldsymbol{x}$ を 1 つ求める問題であり，線形計画問題は，目的関数が線形で制約条件が線形等式や線形不等式で記述される最も基本的な最適化問題である．通常の線形計画問題では，全ての変数は連続的な実数値を取るが，全ての変数が離散的な整数値のみを取る線形計画問題は整数（線形）計画問題と呼ばれる．また，一部の変数が整数値のみを取る場合は混合整数計画問題，全ての変数が $\{0,1\}$ の 2 値のみを取る場合は 0-1 整数計画問題と呼ばれる．最近では非線形の問題も含めて整数計画問題と呼ばれる場合が多いが，本論文では線形の問題のみを整数計画問題と呼ぶ．また，混合整数計画問題や 0-1 整数計画問題も区別せずに整数計画問題と呼ぶ．整数変数は離散的な値を取る事象を表すだけではなく，制約式や状態を切り替えるスイッチとして用いることが可能であり，産業や学術の幅広い分野における現実問題を整数計画問題に定式化できる．組合せ最適化問題は，制約条件を満たす解の集合が組合せ的な構造を持つ最適化問題であり，解が集合，順序，割当て，グラフ，論理値，整数などで記述される場合が多い．原理的に，全ての組合せ最適化問題は整数計画問題に定式化できることが知られており，最近では，整数計画ソルバーの性能向上とも相まって，整数計画ソルバーを用いて組合せ最適化問題を解く事例が増えている．

現実問題を線形計画問題や整数計画問題に定式化する際には，線形式のみを用いて目的関数と制約条件を記述する必要がある．こう書くと，扱える現実問題がかなり限定されるように思われる．実際に，線形計画法の生みの親である Dantzig も Wisconsin 大学で講演をした際に「残念ながら宇宙は線形ではない」と批判を受けている (今野 2005)．しかし，正確さを失うことなく現実問題を非線形計画問題に定式化できても最適解を求められない場合も多く，逆に非線形に見える問題でも変数の追加や式の変形により等価な線形計画問題や整数計画問題に変換できる場合も少なくない．そのため，現実問題を線形計画問題や整数計画問題に定式化してその最適解を求めることは，実用的な問題解決の手法として受け入れられている．

現在では，整数計画ソルバーは現実問題を解決するための有用な道具として数理最適化以外の分野でも急速に普及している．一方で，数理最適化の専門家ではない利用者にとって，線形式のみを用いて現実問題を記述することは容易な作業ではなく，現実問題を上手く定式化できずに悩んだり，強力だが専門家だけが使う良く分からない手法だと敬遠している利用者も少なくない．そこで，本論文では，数理最適化の専門家ではない利用者が，現実問題の解決に取り組む際に必要となる整数計画ソルバーの基本的な利用法と定式化の技法を解説する．なお，最近の整数計画ソルバーはアルゴリズムを知らなくても不自由なく利用できる場合が多いため，本論文では，線形計画法，整数計画法の解法および理論に関する詳しい説明は行わない．線形計画法については (Chvatal 1983; 今野 1987)，整数計画法については (今野 1982; Nemhauser and Wolsey 1988; Wolsey 1998) が詳しい．また，線形計画法，整数計画法の発展の歴史については (Achterberg and Wunderling 2013; Ashford 2007; Bixby and Rothberg 2007; 今野 2005, 2014;

Lodi 2010) が詳しい.

2 線形計画問題と整数計画問題

線形計画問題は，目的関数が線形で制約条件が線形等式や線形不等式で記述される最適化問題であり，一般に以下の標準形で表される.

$$\begin{aligned} \text{minimize} \quad & \sum_{j=1}^{n} c_j x_j \\ \text{subject to} \quad & \sum_{j=1}^{n} a_{ij} x_j \geq b_i, \quad i = 1, \ldots, m, \\ & x_j \geq 0, \qquad\qquad j = 1, \ldots, n. \end{aligned} \tag{1}$$

ここで, a_{ij}, b_i, c_j は定数, x_j は変数である．制約式を全て満たす変数値の組を実行可能解と呼び, 実行可能解全体の集合を実行可能領域と呼ぶ．実行可能解の中で目的関数の値を最小にする解が最適解であり, このときの目的関数の値を最適値と呼ぶ．線形計画問題は $\min\{\boldsymbol{c}^T\boldsymbol{x} \mid \boldsymbol{A}\boldsymbol{x} \geq \boldsymbol{b}, \boldsymbol{x} \geq \boldsymbol{0}\}$ と記述される場合も多い．ここで，各変数の非負条件 $x_j \geq 0$ を（非負）整数条件 $x_j \in \mathbb{Z}_+$（$\mathbb{Z}_+$ は非負整数集合）に置き換えると整数計画問題となる.

線形計画問題では効率の良いアルゴリズムが開発されており，一番最初に単体法[1]が 1947 年に Dantzig によって提案されている．単体法は実用的には優れた性能を持つが，理論的には多項式時間アルゴリズムではない．その後，初めての多項式時間アルゴリズムとなる楕円体法が 1979 年に Khachiyan に，さらに実用的にも高速な内点法が 1984 年に Karmarkar によって提案されている．現在では，単体法と内点法が実用的なアルゴリズムとして広く使われている．性能では内点法の方が優れているが，単体法は制約式や変数を追加して解き直す再最適化を効率良く実行できるため，単体法を用いた再最適化は整数計画問題を解く上で重要な役割を担っている.

整数計画問題は実行可能解の数が有限となる場合が多く，理論上は全ての実行可能解を列挙すれば最適解が求められる．しかし，この種の列挙法は問題の規模の増加とともに走査する解の個数が急激に増加（組合せ的爆発）するため実用的ではない．整数計画問題は NP 困難と呼ばれる問題のクラスに属することが計算の複雑さの理論により知られている．詳しい説明は省略するが，NP 困難問題の最適解を求めようとすると，最悪の場合に全ての実行可能解を列挙するのと本質的に変わらない計算時間が必要であろうと予想されている[2].

整数計画問題では，分枝限定法と切除平面法が代表的なアルゴリズムとして知られている.

[1] 1965 年に Nelder と Mead が非線形計画問題に対して単体法と呼ばれるアルゴリズムを提案しているが，名前が同じというだけで全く異なるアルゴリズムである.

[2] この予想が正しいかどうかは未解決で P $\neq$ NP 予想として有名である.

分枝限定法は，直接解くことが難しい問題をいくつかの小規模な部分問題に分解する分枝操作と，生成された部分問題のうち何らかの理由で最適解が得られないと判定されたものを除く限定操作の 2 つの操作を繰返し適用するアルゴリズムで，整数計画問題以外にも多くの最適化問題で使われている．整数計画問題に対する分枝限定法は 1960 年に Land と Doig によって提案されており，暫定解（これまでの探索で得られた最良の実行可能解）から得られる最適値の上界値と，線形計画緩和問題（各変数の整数条件を緩和して得られる線形計画問題）を解いて得られる最適値の下界値を利用した限定操作で無駄な探索を省くアルゴリズムである．切除平面法は 1958 年に Gomory によって提案されており，線形計画緩和問題から始めて，切除平面（実行可能な整数解を残しつつ線形計画緩和問題の最適解を除去する制約式）を組織的に生成し，線形計画緩和問題に逐次追加することで最終的に整数最適解を得るアルゴリズムである．切除平面法は単体では実用的なアルゴリズムではなく，現在では，分枝限定法の内部で切除平面を逐次追加し，変数値の固定や部分問題に対する下界値の改善を実現する分枝切除法が大きな成功を収めている．

3　整数計画問題の応用事例

これまで，産業や学術の幅広い分野における多くの現実問題が整数計画問題に定式化されてきた．ここでは自然言語処理の応用事例として文書の自動要約 (Filatova and Hatzivassiloglou 2004; Gillick and Favre 2009; 平尾，鈴木，磯崎 2009; McDonald 2007; 西川，平尾，牧野，松尾，松本 2013; 高村，奥村 2008) と文の対応付け (西野，平尾，永田 2013; 西野，鈴木，梅谷，平尾，永田 2014) を紹介する．

3.1　文書の自動要約

文書の自動要約は与えられた単数もしくは複数の文書から要約を生成する問題であり，与えられた文書から必要な文の組合せを選択する手法が知られている．文書要約の問題には，1 つだけの文書が与えられる単一文書の要約と，同じトピックについて記述した複数の文書が与えられる複数文書の要約がある[3]．

まず，単一文書の要約を考える．m 個の概念と n 個の文と要約長 L が与えられる．概念 i の重要度を w_i (>0)，文 j の長さを l_j，文 j に含まれる概念 i の数を $a_{ij} \in \{0,1\}$ と表す．x_j は変数で，文 j が要約に含まれるならば $x_j = 1$，そうでなければ $x_j = 0$ の値を取る．文 j に含まれる概念の重要度の合計 $p_j = \sum_{i=1}^{m} w_i a_{ij}$ はあらかじめ計算できるので，要約長 L を超えない範

[3] 単一の文書に類似した内容の文は含まれないと仮定する．

囲で重要度の合計が最大となる要約を構成する問題は以下の通りに定式化できる．

$$\begin{aligned} \text{maximize} \quad & \sum_{j=1}^{n} p_j x_j \\ \text{subject to} \quad & \sum_{j=1}^{n} l_j x_j \leq L, \\ & x_j \in \{0,1\}, \qquad j = 1, \ldots, n. \end{aligned} \tag{2}$$

この問題はナップサック問題と呼ばれる NP 困難のクラスに属する組合せ最適化問題であるが，動的計画法や分枝限定法に基づく効率良いアルゴリズムが知られている (Kellerer, Pferschy, and Pisinger 2004; Korte and Vygen 2012)．

次に複数文書の要約を考える．複数の文書に類似した内容の文が含まれる場合はこれらの文が同時に選択され，生成された要約の中に類似した内容が繰返し現れる恐れがある．そこで，概念 i が要約に含まれているならば $z_i = 1$，そうでなければ $z_i = 0$ の値を取る変数 z_i を導入すると以下の通りに定式化できる．

$$\begin{aligned} \text{maximize} \quad & \sum_{i=1}^{m} w_i z_i \\ \text{subject to} \quad & \sum_{j=1}^{n} a_{ij} x_j \geq z_i, \quad i = 1, \ldots, m, \\ & \sum_{j=1}^{n} l_j x_j \leq L, \\ & x_j \in \{0,1\}, \qquad j = 1, \ldots, n, \\ & z_i \in \{0,1\}, \qquad i = 1, \ldots, m. \end{aligned} \tag{3}$$

1 番目の制約条件は，左辺の値に関わらず $z_i = 0$ の値を取れば必ず制約条件が満たされるため，最適解において概念 i を含む文 j が要約に含まれているにも関わらず $z_i = 0$ の値を取る場合があるように思われる．しかし，目的関数は最大化で各変数 z_i の係数 w_i は正の値であり，このような場合には $z_i = 1$ の値を取れば改善解が得られるため，最適解では概念 i を含む文 j が要約に含まれていれば必ず $z_i = 1$ の値を取ることが分かる．この定式化では，重要度の高い概念が要約の中に繰返し現れても目的関数は増加しないので，冗長性を自然に抑えることができる．一方で，この問題は（ナップサック制約付き）最大被覆問題と呼ばれる NP 困難のクラスに属する組合せ最適化問題であり，大規模な問題例では最適解を効率良く求めることは難しい．複数文書の要約を求める問題は，この他にも施設配置問題 (高村，奥村 2010) や冗長制約付きナップサック問題 (西川 他 2013) などに定式化されている．

3.2　文の対応付け

統計的機械翻訳では，対訳コーパスにおいて原言語文と目的言語文の対応付けが与えられている前提の下で処理が適用される．しかし，実際の対訳コーパスでは，文書同士の対応付けは行われていても，それらの文書に含まれる文同士の対応付けは行われていない場合が多い．そのため，対訳文書の間で文同士の正しい対応付けを求めることは，統計的機械翻訳の精度を上げるための重要な前処理となる．

(Ma 2006; Moore 2002) などは，対訳文書間で対応する文の出現順序が大きく入れ替わらないという前提で動的計画法に基づく文の対応付けを提案している．すなわち，対訳文書の組 F，E が与えられたとき，F の i 番目の文と E の j 番目の文が対応するならば，F の $i+1$ 番目の文に対応する E の文は，（存在するならば）j 番目の近くにあるという前提で文の対応付けを行っている．しかし，文の出現順序が大きく入れ替わらないという前提はどの文書でも成り立つ性質ではない．

まず，文書 F と E の任意の文を出現順序に関わらず自由に対応付けても良い場合を考える．n_f 個の文を含む文書 F と n_e 個の文を含む文書 E が与えられる．文書 F の i 番目の文と文書 E の j 番目の文が対応付けられたときのスコアを s_{ij} と表す．x_{ij} は変数で，文書 F の i 番目の文と文書 E の j 番目の文が対応付けられるならば $x_{ij}=1$，そうでなければ $x_{ij}=0$ の値を取る．このとき，文書 F の文は文書 E の高々 1 つの文にしか対応付けられない（その逆も同様）という制約を課すと，スコアの合計が最大となる文の対応付けを求める問題は以下の通りに定式化できる．

$$
\begin{aligned}
\text{maximize} \quad & \sum_{i=1}^{n_f}\sum_{j=1}^{n_e} s_{ij}x_{ij} \\
\text{subject to} \quad & \sum_{j=1}^{n_e} x_{ij} \le 1, \quad i=1,\ldots,n_f, \\
& \sum_{i=1}^{n_f} x_{ij} \le 1, \quad j=1,\ldots n_e, \\
& x_{ij} \in \{0,1\}, \quad i=1,\ldots,n_f,\ j=1,\ldots,n_e.
\end{aligned}
\tag{4}
$$

この問題は（2 部グラフの）最大重みマッチング問題と呼ばれる組合せ最適化問題であり，ハンガリー法など効率良いアルゴリズムが知られている (Korte and Vygen 2012).

文書 F と E の任意の文を対応付けても良いという前提では，それぞれの文書において前後の文との繋がりを無視した対応付けを行うことになるため，段落のような文の系列（連続する文のまとまり）を単位として順序が入れ替わる場合には正しい順序付けができない可能性が高い．そこで，図 1 に示すように，文書 F と文書 E をそれぞれ同数の文の系列に分割し，文の系列について一対一の対応付けを求めることを考える．文書 F の i 番目から j 番目までの文の系列を

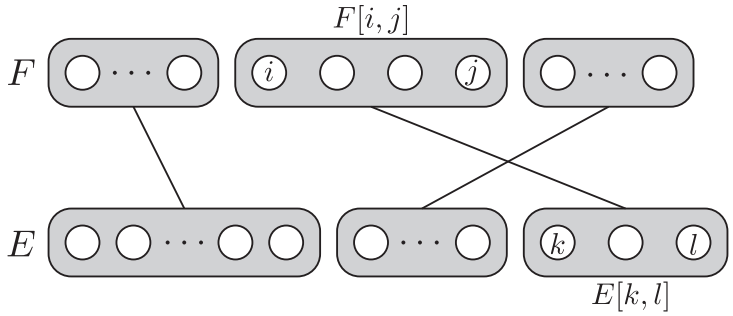

図 1　出現順序の入れ替わりを考慮した文の系列の対応付け

$F[i,j](1 \le i \le j \le n_f)$，文書 E の k 番目から l 番目までの文の系列を $E[k,l](1 \le k \le l \le n_e)$ とする．文 $p \in F[i,j]$ の文 $q \in E[k,l]$ の対応付けでは出現順序の入れ替わりはないとすると，(Moore 2002) の手法を適用することで文 $p \in F[i,j]$ と文 $q \in E[k,l]$ の最適な対応付けとスコアの合計 w_{ijkl} を計算できる．考えられ得る全ての文の系列の組 $(F[i,j], E[k,l])$ に対して，それらの系列に含まれる文同士の最適な対応付けとスコアの合計 w_{ijkl} をあらかじめ計算する．x_{ijkl} は変数で，文の系列 $F[i,j]$ と $E[k,l]$ が対応付けられると $x_{ijkl} = 1$，そうでなければ $x_{ijkl} = 0$ の値を取る．各文 $p \in F$, $q \in E$ が対応付けられたいずれかの文の系列 $F[i,j]$, $E[k,l]$ にちょうど 1 回ずつ含まれるという制約条件を用いて文の系列を一対一に対応付ける．このとき，スコアの合計が最大となる文の系列の対応付けを求める問題は以下の通りに定式化できる．

$$
\begin{aligned}
\text{maximize} \quad & \sum_{1 \le i \le j \le n_f} \sum_{1 \le k \le l \le n_e} w_{ijkl} x_{ijkl} \\
\text{subject to} \quad & \sum_{i \le p \le j} \sum_{1 \le k \le l \le n_e} x_{ijkl} = 1, && p = 1, \ldots, n_f, \\
& \sum_{1 \le i \le j \le n_f} \sum_{k \le q \le l} x_{ijkl} = 1, && q = 1, \ldots, n_e, \\
& x_{ijkl} \in \{0, 1\}, && 1 \le i \le j \le n_f,\ 1 \le k \le l \le n_e.
\end{aligned}
\tag{5}
$$

この問題は集合分割問題と呼ばれる NP 困難のクラスに属する組合せ最適化問題であり，大規模な問題例では最適解を効率良く求めることは難しい．

統計的機械翻訳では，この他にもフレーズ（連続する単語列）の対応付けを求める問題が整数計画問題に定式化されている (DeNero and Klein 2008; 越川，内山，梅谷，松井，山本 2010)．

4　整数計画ソルバーを利用する

前節で紹介したように，実際に多くの現実問題が整数計画問題として定式化できる．一方で，整数計画問題を含む多くの組合せ最適化問題は NP 困難のクラスに属することが計算の複雑さ

の理論により明らかにされている．こう書くと，多くの現実問題に対して最適解を求めることは非常に困難であるように思われるが，計算の複雑さが示す結果の多くは「最悪の場合」であり，全ての入力データに対して最適解を求めることは困難でも，多くの入力データに対して現実的な計算時間で最適解を求められる問題は少なくない．また，整数計画ソルバーは探索中に得られた暫定解を保持しているので，与えられた計算時間内に最適解が求められなくても，精度の高い実行可能解が求まれば，利用者によっては十分に満足できる場合も多く，整数計画ソルバーはそのような目的にも使われる．

表 1 に示すように，現在では，商用・非商用を含めて多数の整数計画ソルバーが利用可能である (Atamtürk and Savelsbergh 2005; Linderoth and Ralph 2006; Fourer 2013; Mittelmann 2014)．商用ソルバーを利用するためには，数十万〜数百万円のライセンス料金が必要となる場合が多いが，無償の試用ライセンスや無償〜数十万円のアカデミックライセンスが用意されている場合も少なくない．一般的に，非商用ソルバーより商用ソルバーの方が性能は高いが，実際には商用ソルバーの中でもかなりの性能差がある．整数計画ソルバーのベンチマーク問題例に対する最新の実験結果 (Mittelmann 2014) によると，商用ソルバーでは，先に挙げた Xpress Optimization Suite, Gurobi Optimizer, CPLEX Optimization Studio の 3 つが，非商用ソルバーでは SCIP が最も性能が高いようである．整数計画ソルバーを選ぶ際には，性能以外にも，扱える問題の種類[4]，扱える問題の記述形式，インターフェースなどを考慮して，各自の目的に合った整数計画ソルバーを選ぶことが望ましい．利用可能な整数計画ソルバーについては (Fourer 2013; Mittelmann

表 1　代表的な整数計画ソルバー

	名称	販売・公開元
商用	CPLEX Optimization Suite	IBM Corporation
	Gurobi Optimizer	Gurobi Optimization, Inc.
	LINDO API	LINDO Systems, Inc.
	MOSEK	MOSEK ApS.
	Numerical Optimizer	NTT DATA Mathematical Systems, Inc.
	SAS	SAS Institute, Inc.
	SOPT	SAITECH, Inc.
	Xpress Optimization Suite	Fair Isaac Corporation
非商用	CBC	COIN-OR Foundation
	GLPK	Free Software Foundation, Inc.
	lp_solve	Open Source
	SCIP	Zuse Institute Berlin
	SYMPHONY	COIN-OR Foundation

[4] 最近では非線形の整数計画問題を扱えるソルバーも増えている．

2014) が詳しい.

まず，整数計画ソルバーを用いて以下の問題例を解くことを考える.

$$\begin{aligned} \text{maximize} \quad & 2x_1 + 3x_2 \\ \text{subject to} \quad & 2x_1 + x_2 \leq 10, \\ & 3x_1 + 6x_2 \leq 40, \\ & x_1, x_2 \in \mathbb{Z}_+. \end{aligned} \tag{6}$$

整数計画ソルバーの主な利用法には，(1) コマンドラインインターフェースを通じてソルバーを実行する方法，(2) 最適化モデリングツールを通じてソルバーを実行する方法，(3) 他のソフトウェアから API[5]を通じてソルバーを実行する方法の 3 通りがある.

1 番目は，問題例を LP 形式[6]，MPS 形式[7]などで記述された入力ファイルを用意して整数計画ソルバーを実行する方法である．図 2 は問題例 (6) を LP 形式で記述したものである．目的関数や制約条件の部分は，数式をほぼそのまま記述しているだけである[8]．`maximize`, `subject to`, `bounds`, `general`, `end` は予約語で，変数値の上下限や整数制約などをこれらの予約語を用いて記述している．LP 形式は文法が平易で可読性が高く，多くの整数計画ソルバーが対応している．図 3 は問題例 (6) を MPS 形式で記述したものである．MPS 形式は 1960 年代に IBM によって導入された形式で，現在も標準的に使われているが可読性は低い．LP 形式や MPS 形式はプログラミング言語の配列のように変数をまとめて扱う記述ができない．つまり，LP 形式や MPS 形式で $\sum_{j=1}^{100} x_j \leq 3$ の数式を記述するには，`x1 + x2 +（中略）+ x100 <= 3` と書くしか

```
maximize
 obj: 2 x1 + 3 x2
subject to
 c1: 2 x1 +   x2 <= 10
 c2: 3 x1 + 6 x3 <= 40
bounds
 x1 >= 0
 x2 >= 0
general
 x1 x2
end
```

図 2　問題例 (6) の LP 形式による記述

[5] API は “Application Programming Interface” の略.
[6] LP は “Linear Programming” の略.
[7] MPS は “Mathematical Programming System” の略.
[8] 図 2 では非負制約を記述しているが，LP 形式では何も指定しなければ各変数 x_j の非負制約 $x_j \geq 0$ は自動的に設定される.

```
NAME          sample
ROWS
 N  obj
 L  c1
 L  c2
COLUMNS
    INTSTART  'MARKER'           'INTORG'
    x1        obj           -2   c1             2
    x1        c2             3
    x2        obj           -3   c1             1
    x2        c2             6
    INTEND    'MARKER'           'INTEND'
RHS
    RHS       c1            10   c2            40
BOUNDS
 PL Bound     x1
 PL Bound     x2
ENDATA
```

図 3 問題例 (6) の MPS 形式による記述

方法がない．よって，大きな問題例を記述する場合には，適当なプログラム言語を用いて LP 形式や MPS 形式のファイルを生成するプログラムを作成する必要がある．

2 番目は，最適化モデリングツールが提供するモデリング言語で問題例を記述し，最適化モデリングツールを通じて整数計画ソルバーを実行する方法である．商用の最適化モデリングツールが提供するモデリング言語では，AIMMS, AMPL, GAMS など，非商用では，Math Prog, ZIMPL などが知られている．図 4 は問題例 (6) を Math Prog 形式で記述したものである．多くのモデリング言語では，モデル部分とデータ部分を分離して記述できるため，数式を直感的にモデルに書き換えることが可能である．例えば，$\sum_{j=1}^{100} x_j \leq 3$ の数式は，`sum(i in 1..100)(x[i]) <= 3` と記述できる．現実問題を最適化問題に定式化できればすぐに整数計画ソルバーを利用できるので効率良いプロトタイピングが可能となる．一方で，最適化モデリングツールの購入とモデリング言語の習得が必要で，1 番目の方法に比べると汎用性に欠ける．

3 番目は，整数計画ソルバーが提供する C, C++, Java, Python, Matlab, Excel などのライブラリやプラグインを通じて整数計画ソルバーを実行する方法である．部分問題を解くためのサブルーチンとして整数計画ソルバーを利用する場合や，整数計画ソルバーの挙動を細かく制御したい場合はこの方法が効率的である．ただし，整数計画ソルバーやそのバージョン毎にライブラリやプラグインの仕様が異なるため汎用性と保守性に欠ける．

最適化ソルバーの利用者にとって，与えられた問題例がどの程度の計算時間で解けるかを事前に見積ることは重要である．線形計画問題では，一部の特殊な問題を除けば変数や制約式の数を計算時間の目安にして差し支えない場合が多い．一方で，整数計画問題では，(Koch, Achter-

```
param n, integer;
param m, integer;
param c{j in 1..n};
param a{i in 1..m, j in 1..n};
param b{i in 1..m};
var x{j in 1..n}, integer >= 0;

minimize z: sum{j in 1..n} c[j]*x[j];
s.t. con{i in 1..m}: sum{j in 1..n} a[i,j]*x[j] <= b[i];

data;
param n := 2;
param m := 2;
param c := 1 -2, 2 -3;
param a: 1 2 :=
      1  2 1
      2  3 6;
param b := 1 10, 2 40;
end;
```

図 4 問題例 (6) の Math Prog 形式による記述

berg, Andersen, Bastert, Berthold, Bixby, Danna, Gamrath, Gleixner, Heinz, Lodi, Mittelmann, Ralphs, Salvagnin, Steffy, and Wolter 2011) で報告されているように，10 万変数，10 万制約式で最適解を効率良く求められる問題例がある一方で，1,000 変数程度でも最適解を求められない問題例があり，変数や制約式の数だけでは計算時間を見積れないことが知られている．

整数計画ソルバーの現状や利用法については (Berthold, Gleixner, Heinz, Koch, and Shinano 2012; 藤江 2011; 宮代 2014, 2012; 宮代，松井 2006) が詳しい．

5 線形計画問題に定式化する

線形計画問題では，数百万変数，数百万制約式の大規模な問題例でも現実的な計算時間で最適解を求められるが，線形式のみを用いて目的関数と制約条件を記述する必要があるため，数理最適化の専門家ではない利用者にとって，現実問題を線形計画問題に定式化することは容易な作業ではない．しかし，一見すると非線形計画問題に見える問題も変数の追加や式の変形により等価な線形計画問題に変換できる場合は少なくない．現実問題を線形計画問題を定式化する際には，与えられた現実問題を線形計画問題で正確に記述できるか，または満足できる程度に近似できるか良く見極める必要がある．ここでは，一見すると非線形に見える最適化問題を線形計画問題に定式化するいくつかの方法を紹介する．

5.1　凸関数最小化問題

凸関数最小化問題は線形計画問題に近似できる．ここでは，図 5 に示すように，1 変数の凸関数 $f(x)$ を区分線形関数 $g(x)$ で近似する方法を考える．凸関数 $f(x)$ 上の m 個の点 $(a_1, f(a_1))$, $\ldots, (a_m, f(a_m))$ を適当に選んで線分で繋ぐと区分線形関数 $g(x)$ が得られる．この区分線形関数 $g(x)$ は凸関数なので，各区分を表す線形関数を用いて，

$$g(x) = \max_{i=1,\ldots,m-1} \left\{ \frac{f(a_{i+1}) - f(a_i)}{a_{i+1} - a_i}(x - a_i) + f(a_i) \right\}, \quad a_1 \leq x \leq a_m, \tag{7}$$

と記述できる．このとき，各区分を表す線形関数の最大値を表す変数 z を用意すると，区分線形関数 $g(x)$ の最小化問題は以下の線形計画問題に定式化できる．

$$\begin{aligned} &\text{minimize} && z \\ &\text{subject to} && \frac{f(a_{i+1}) - f(a_i)}{a_{i+1} - a_i}(x - a_i) + f(a_i) \leq z, \quad i = 1, \ldots, m-1, \\ & && a_1 \leq x \leq a_m. \end{aligned} \tag{8}$$

多変数の凸関数最小化問題でも，直線の集合の代わりに超平面の集合で凸関数を近似すれば同様に定式化できる．

5.2　連立 1 次方程式の近似解

全ての制約式を同時には満たせない連立 1 次方程式に対して，できる限り多くの制約式を満たす近似解を求める問題は目標計画法と呼ばれる (Charnes and Cooper 1961)．連立 1 次方程式

$$\sum_{j=1}^{n} a_{ij} x_j = b_i, \quad i = 1, \ldots, m, \tag{9}$$

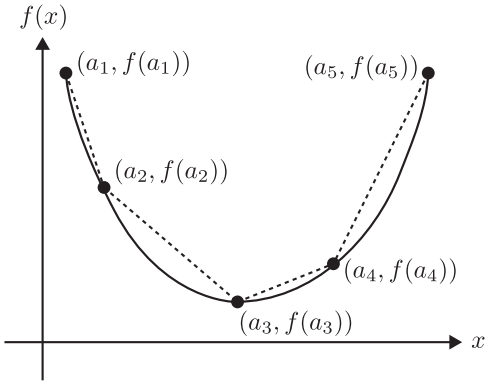

図 5　凸関数の区分線形関数による近似

に対して，その誤差

$$z_i = \left| b_i - \sum_{j=1}^{n} a_{ij} x_j^* \right|, \quad i = 1, \ldots, m, \tag{10}$$

をできる限り小さくする近似解 $\boldsymbol{x}^* = (x_1^*, \ldots, x_n^*)$ を求める問題を考える．このとき，平均 2 乗誤差 $\frac{1}{m}\sum_{i=1}^{m} z_i^2$，平均誤差 $\frac{1}{m}\sum_{i=1}^{m} z_i$，最悪誤差 $\max_{i=1,\ldots,m} z_i$ などが評価基準として考えられる．これらの評価基準は，それぞれ誤差ベクトル $\boldsymbol{z} = (z_1, \ldots, z_m)$ の L_2 ノルム，L_1 ノルム，L_∞ ノルムを最小化する近似解 $\boldsymbol{x}^*$ を求めることに対応する．応用事例では，誤差の分布がガウス分布に近いことを前提に，平均 2 乗誤差を評価基準として最小 2 乗法を用いて近似解を求める場合が多い．しかし，実際には外れ値が多いなど誤差の分布がガウス分布と全く異なる場合も少なくない．このような場合は，外れ値の影響を受けにくい平均誤差や最悪誤差を評価基準として近似解を求める方法が考えられる．

平均誤差を最小化する近似解 $\boldsymbol{x}^*$ を求める問題は，制約式を持たない以下の最適化問題に定式化できる．

$$\text{minimize} \quad \sum_{i=1}^{m} \left| b_i - \sum_{j=1}^{n} a_{ij} x_j \right|. \tag{11}$$

これは一見しただけでは線形計画問題に見えないが以下の線形計画問題に変換できる．

$$\begin{aligned} \text{minimize} \quad & \sum_{i=1}^{m} z_i & & \\ \text{subject to} \quad & b_i - \sum_{j=1}^{n} a_{ij} x_j \geq -z_i, & i &= 1, \ldots, m, \\ & b_i - \sum_{j=1}^{n} a_{ij} x_j \leq z_i, & i &= 1, \ldots, m, \\ & z_i \geq 0, & i &= 1, \ldots, m. \end{aligned} \tag{12}$$

この方法で線形回帰問題を解くこともできる．m 個のデータ $(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_m, y_m)$ が与えられる．これを n 個の関数 $\{\phi_1(\boldsymbol{x}_i), \ldots, \phi_n(\boldsymbol{x}_i)\}$ の線形結合を用いて $y(\boldsymbol{x}_i) \approx w_0 + w_1\phi_1(\boldsymbol{x}_i) + \cdots + w_n\phi_n(\boldsymbol{x}_i)$ と近似する問題を考える．各データ $(\boldsymbol{x}_i, y_i)$ に対する平均誤差を最小にするパラメータ $w_0, \ldots, w_n$ を求める問題は以下の最適化問題に定式化できる．

$$\text{minimize} \quad \sum_{i=1}^{m} \left| y_i - (w_0 + w_1\phi_1(\boldsymbol{x}_i) + \cdots + w_n\phi_n(\boldsymbol{x}_i)) \right| \tag{13}$$

i 番目のデータに対する誤差を表す変数 z_i を用意すると以下の線形計画問題[9]に定式化できる．

[9] $\boldsymbol{x}_i$ は与えられたデータなので $\phi_1(\boldsymbol{x}_i), \ldots, \phi_n(\boldsymbol{x}_i)$ は定数となる．

$$
\begin{aligned}
\text{minimize} \quad & \sum_{i=1}^{m} z_i \\
\text{subject to} \quad & y_i - (w_0 + w_1\phi_1(\boldsymbol{x}_i) + \cdots + w_n\phi_n(\boldsymbol{x}_i)) \geq -z_i, \quad i = 1, \ldots, m, \\
& y_i - (w_0 + w_1\phi_1(\boldsymbol{x}_i) + \cdots + w_n\phi_n(\boldsymbol{x}_i)) \leq z_i, \quad i = 1, \ldots, m, \\
& z_i \geq 0, \quad i = 1, \ldots, m.
\end{aligned} \tag{14}
$$

最悪誤差を最小化する近似解 $\boldsymbol{x}^*$ を求める問題も制約式を持たない以下の最適化問題に定式化できる．

$$
\text{minimize} \quad \max_{i=1,\ldots,m} \left| b_i - \sum_{j=1}^{n} a_{ij} x_j \right|. \tag{15}
$$

これも一見すると線形計画問題に見えないが以下の線形計画問題に変換できる．

$$
\begin{aligned}
\text{minimize} \quad & z \\
\text{subject to} \quad & b_i - \sum_{j=1}^{n} a_{ij} x_j \geq -z, \quad i = 1, \ldots, m, \\
& b_i - \sum_{j=1}^{n} a_{ij} x_j \leq z, \quad i = 1, \ldots, m, \\
& z \geq 0.
\end{aligned} \tag{16}
$$

この方法で k 個の目的関数 $\sum_{j=1}^{n} c_{1j}x_j, \sum_{j=1}^{n} c_{2j}x_j, \ldots, \sum_{j=1}^{n} c_{kj}x_j$ を同時に最小化する多目的最適化問題も解くことができる．

$$
\begin{aligned}
\text{minimize} \quad & \left\{ \sum_{j=1}^{n} c_{1j}x_j, \ldots, \sum_{j=1}^{n} c_{kj}x_j \right\} \\
\text{subject to} \quad & \sum_{j=1}^{n} a_{ij} x_j \geq b_i, \quad i = 1, \ldots, m, \\
& x_j \geq 0, \quad j = 1, \ldots, n.
\end{aligned} \tag{17}
$$

まず，これらの目的関数の線形和を最小化する定式化が考えられる．しかし，いくつかの目的関数が極端に大きな値となる解が求まってしまう場合が少なくないため，全ての目的関数をバランス良く最小化することは容易ではない．そこで，これらの目的関数の最大値を最小化する定式化を考える．新たに目的関数の最大値を表す変数 z を導入すると，この問題は以下の線形計画問題に変換できる．

$$\begin{aligned} \text{minimize} \quad & z \\ \text{subject to} \quad & \sum_{j=1}^{n} c_{hj} x_j \le z, \quad h = 1, \ldots, k, \\ & \sum_{j=1}^{n} a_{ij} x_j \ge b_i, \quad i = 1, \ldots, m, \\ & x_j \ge 0, \qquad\qquad j = 1, \ldots, n. \end{aligned} \tag{18}$$

5.3　比率の最小化

2 つの関数の比を目的関数に持つ最適化問題は分数計画問題と呼ばれる．以下の 2 つの線形関数の比を目的関数に持つ分数計画問題を考える．

$$\begin{aligned} \text{minimize} \quad & \frac{\sum_{j=1}^{n} c_j x_j}{\sum_{j=1}^{n} d_j x_j} \\ \text{subject to} \quad & \sum_{j=1}^{n} a_{ij} x_j = b_i, \quad i = 1, \ldots, m. \end{aligned} \tag{19}$$

ただし，$\sum_{j=1}^{n} d_j x_j > 0$ とする．ここで，新たな変数 $t = 1/\sum_{j=1}^{n} d_j x_j$ と $y_j = t x_j$ $(j = 1, \ldots, n)$ を導入すると，この問題は以下の線形計画問題に変換できる．

$$\begin{aligned} \text{minimize} \quad & \sum_{j=1}^{n} c_j y_j \\ \text{subject to} \quad & \sum_{j=1}^{n} a_{ij} y_j - b_i t = 0, \quad i = 1, \ldots, m, \\ & \sum_{j=1}^{n} d_j y_j = 1. \end{aligned} \tag{20}$$

この変換は $\sum_{j=1}^{n} d_j x_j$ が常に同じ符号で 0 にならない場合のみ成立するので，必要があれば $\sum_{j=1}^{n} d_j x_j \ge \varepsilon$ （ε は十分に小さな正の定数）などの制約式を追加すれば良い．

6　整数計画問題に定式化する

整数計画問題は整数変数を含む線形計画問題であるが，線形計画問題の方が整数計画問題よりもはるかに解き易い事実を考慮すれば，現実問題において離散値を取る量を決定するという理由だけで安易に整数変数を用いるべきではない．例えば，自動車や機械部品の生産数を決定する問題を整数計画問題に定式化することは必ずしも適切ではない．このような場合は，各変数の整数条件を取り除いた線形計画問題を解いて実数最適解を得た後に，その端数を丸めて最

も近い整数解を求めれば十分に実用的な解となる場合が多い．実際に，多くの現実的な整数計画問題では，yes/no の決定や離散的な状態の切り替えを記述するために 2 値変数（$\{0,1\}$ の 2 値のみを取る整数変数）を用いていることに注意する必要がある．

ここでは，代表的な組合せ最適化問題を例に整数計画問題の基本的な定式化の技法を紹介する．もちろん，いくつかの組合せ最適化問題では効率良いアルゴリズムが知られているが，現実問題が既知の組合せ最適化問題と一致することは稀であり，これらの効率良いアルゴリズムをそのまま適用できるとは限らない．一方で，整数計画ソルバーであれば定式化を少し変形するだけで適用できる場合が多い．このように，代表的な組合せ最適化問題に対する整数計画問題の定式化を知れば，それらを雛形として変形もしくは組合せることで多種多様な現実問題を整数計画問題に定式化できるようになる．

6.1　整数性を持つ整数計画問題

整数計画問題は一般には NP 困難のクラスに属する計算困難な問題であるが，いくつかの特殊な整数計画問題は効率良く解けることが知られている．ここでは，制約行列 $\boldsymbol{A}$ が完全単摸行列である整数計画問題 $\min\{\boldsymbol{c}^T\boldsymbol{x} \mid \boldsymbol{A}\boldsymbol{x} \geq \boldsymbol{b}, \boldsymbol{x} \in \mathbb{Z}_+^n\}$ を紹介する．任意の小行列式が 0, 1, -1 のどれかに等しい行列 $\boldsymbol{A}$ は完全単摸行列と呼ばれる．$\boldsymbol{A}$ が整数行列，$\boldsymbol{b}$ が整数ベクトルである線形計画問題 $\min\{\boldsymbol{c}^T\boldsymbol{x} \mid \boldsymbol{A}\boldsymbol{x} \geq \boldsymbol{b}, \boldsymbol{x} \geq \boldsymbol{0}\}$ について，$\boldsymbol{A}$ が完全単摸行列で（実数）最適解が存在するならば，単体法を適用すると常に整数最適解 $\boldsymbol{x} \in \mathbb{Z}_+^n$ が得られる．

有向グラフが与えられたとき，点の番号を行番号，辺の番号を列番号とする行列 $\boldsymbol{A}$ で，辺 $e = (i,j)$ に対応する列が $a_{ie} = 1$, $a_{je} = -1$（その他は 0）で与えられる行列は接続行列と呼ばれる．任意の有向グラフに対して，その接続行列は完全単摸行列となる．また，無向グラフの接続行列では辺に対応する列が $a_{ie} = a_{je} = 1$（その他は 0）で与えられる．無向グラフでは 2 部グラフであるときに限り，その接続行列は完全単摸行列となる．

以下では，完全単摸行列を制約行列に持つ整数計画問題の例として最短路問題と割当問題を紹介する．

最短路問題：有向グラフ $G = (V, E)$ と各辺 $(i,j) \in E$ の長さ d_{ij} が与えられる．x_{ij} は変数で，辺 $(i,j) \in E$ が経路に含まれるならば $x_{ij} = 1$，そうでなければ $x_{ij} = 0$ の値を取る．このとき，与えられた始点 $s \in V$ から終点 $t \in V$ に至る最短路を求める問題は以下の通りに定式化できる．

$$
\begin{aligned}
\text{minimize} \quad & \sum_{(i,j)\in E} d_{ij}x_{ij} \\
\text{subject to} \quad & \sum_{j:(s,j)\in E} x_{sj} = 1, \\
& \sum_{i:(i,t)\in E} x_{it} = 1, \\
& \sum_{i:(i,k)\in E} x_{ik} - \sum_{j:(k,j)\in E} x_{kj} = 0, \quad k \in V \setminus \{s,t\}, \\
& x_{ij} \in \{0,1\}, \qquad\qquad (i,j) \in E.
\end{aligned}
\tag{21}
$$

1 番目と 2 番目の制約式は，始点 s から出る辺と終点 t に入る辺がちょうど 1 本ずつ選ばれることを表す．3 番目の制約式は，訪問する頂点 k では出る辺と入る辺がちょうど 1 本ずつ選ばれ，それ以外の頂点では辺は選ばれないことを表す．

割当問題：m 人の学生を n 個のクラスに割り当てる．クラス j の受講者数の下限を l_j，上限を u_j，学生 i のクラス j に対する満足度を p_{ij} とする．x_{ij} は変数で，学生 i がクラス j に割当てられれば $x_{ij} = 1$，そうでなければ $x_{ij} = 0$ の値を取る．このとき，学生の満足度の合計が最大となる割当てを求める問題は以下の通りに定式化できる．

$$
\begin{aligned}
\text{maximize} \quad & \sum_{i=1}^{m}\sum_{j=1}^{n} p_{ij}x_{ij} \\
\text{subject to} \quad & \sum_{j=1}^{n} x_{ij} = 1, \qquad i = 1,\ldots,m, \\
& l_j \le \sum_{i=1}^{m} x_{ij} \le u_j, \quad j = 1,\ldots,n, \\
& x_{ij} \in \{0,1\}, \qquad i = 1,\ldots,m,\ j = 1,\ldots,n.
\end{aligned}
\tag{22}
$$

1 番目の制約式は，各学生 i がちょうど 1 つのクラスに割当てられることを表す．2 番目の制約式は，各クラス j に割当てられる学生の数が受講者数の上下限内に収まることを表す．

最短路問題，割当問題はそれぞれダイクストラ法やハンガリー法など効率良いアルゴリズムが知られている (Korte and Vygen 2012)．しかし，現実問題では実務上の要求から生じる制約条件が追加される場合が多いため，これらの効率良いアルゴリズムがそのまま適用できるとは限らない．一方で，与えられた現実問題を完全単摸行列に近い形の制約行列を持つ整数計画問題に定式化できる場合は，線形計画緩和問題から良い下界値が得られることが期待できるため，整数計画ソルバーを用いて現実的な計算時間で最適解を求められる場合は少なくない．完全単摸行列の性質については (Korte and Vygen 2012; Schrijver 1998) が詳しい．

6.2 論理的な制約条件

現実問題が既知の組合せ最適化問題と一致することは稀であり，実務上の要求から生じる制約条件が追加される場合が多い．ここでは，ナップサック問題を例にいくつかの論理的な制約条件とその記述を紹介する．

ナップサック問題：1 つの箱と n 個の荷物が与えられる．箱に詰込める重さ合計の上限を $c(>0)$，各荷物 j の重さを $w_j(<c)$，価値を p_j とする．x_j は変数で，荷物 j を箱に詰めるならば $x_j = 1$，そうでなければ $x_j = 0$ の値を取る．このとき，価値の合計が最大となる荷物の詰込みを求める問題は以下の通り定式化できる．

$$\begin{aligned} \text{maximize} \quad & \sum_{j=1}^{n} p_j x_j \\ \text{subject to} \quad & \sum_{j=1}^{n} w_j x_j \leq c, \\ & x_j \in \{0,1\}, \qquad j = 1, \ldots, n. \end{aligned} \tag{23}$$

ちなみに，複数の制約式を持つナップサック問題は多制約ナップサック問題と呼ばれ，投資計画やポートフォリオ最適化などの応用を持つ．ナップサック問題については (Kellerer et al. 2004) が詳しい．

以下に，いくつかの論理的な制約条件とその記述を示す．

(1) 詰込む荷物の数は高々 k 個.

$$\sum_{j=1}^{n} x_j \leq k. \tag{24}$$

(2) 荷物 j_1, j_2 の少なくとも一方は詰込む.

$$x_{j_1} + x_{j_2} \geq 1. \tag{25}$$

(3) 荷物 j_1 を詰込むならば荷物 j_2 も詰込む.

$$x_{j_1} \leq x_{j_2}. \tag{26}$$

(4) 詰込む荷物の数は 0 または 2.

$$\sum_{j=1}^{n} x_j = 2y, \; y \in \{0,1\}. \tag{27}$$

もしくは y を使わずに，以下の通りにも記述できる．

$$\begin{cases} +x_1 + x_2 + \cdots + x_n \leq 2, \\ -x_1 + x_2 + \cdots + x_n \geq 0, \\ +x_1 - x_2 + \cdots + x_n \geq 0, \\ \cdots, \\ +x_1 + x_2 + \cdots - x_n \geq 0. \end{cases} \tag{28}$$

2 番目以降の制約式は $\sum_{j=1}^{n} x_j = 1$ を満たす解を除外しており，図 6 に示すように，これらの制約式は実行可能解全体の凸包（全ての実行可能解を含む最小の凸多面体）から得られる．

6.3　固定費用付き目的関数

生産計画や物流計画など多くの現実問題では，取り扱う製品量によって生じる変動費用と段取替えなど所定の作業によって生じる固定費用の両方を考慮する場合が多い．例えば，x を単位費用 c_1 で生産される製品の生産量とする．もし，その製品が少しでも生産されれば初期費用 c_2 が生じるとすると，総費用 $f(x)$ は以下に示す非線形関数となる（C は製品の生産量の上限とする）．

$$f(x) = \begin{cases} 0 & x = 0 \\ c_1 x + c_2 & 0 < x \leq C. \end{cases} \tag{29}$$

そこで，少しでも製品を生産するならば $y = 1$，そうでなければ $y = 0$ の値を取る 2 値変数 y を

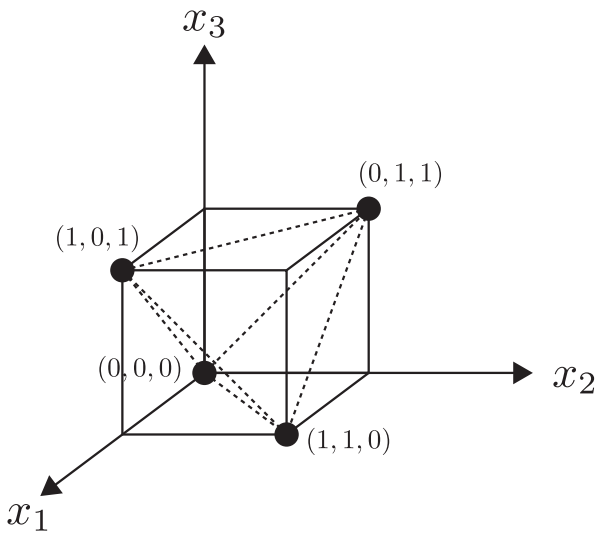

図 6　$x_1 + x_2 + x_3 = 0$ または 2 を満たす全ての解を含む凸包

導入すると，総費用 $f(x)$ は以下の通りに記述できる．

$$f(x) = \{c_1 x + c_2 y \mid x \leq Cy, 0 \leq x \leq C, y \in \{0,1\}\}. \tag{30}$$

以下では，固定費用を持つ整数計画問題の例としてビンパッキング問題を紹介する．

ビンパッキング問題：十分な数の箱と n 個の荷物が与えられる．箱に詰込める荷物の重さ合計の上限を $c(>0)$，各荷物 j の重さを $w_j(<c)$ とする．x_{ij} と y_i は変数で，荷物 j が箱 i に入っていれば $x_{ij}=1$，そうでなければ $x_{ij}=0$，箱 i を使用していれば $y_i=1$，そうでなければ $y_i=0$ の値を取る．このとき，使用する箱の数が最小となる荷物の詰込みを求める問題は以下の通りに定式化できる．

$$\begin{aligned}
\text{minimize} \quad & \sum_{i=1}^{n} y_i \\
\text{subject to} \quad & \sum_{j=1}^{n} w_j x_{ij} \leq c y_i, \quad i = 1, \ldots, n, \\
& \sum_{i=1}^{n} x_{ij} = 1, \qquad\quad j = 1, \ldots, n, \\
& x_{ij} \in \{0,1\}, \qquad\quad i = 1, \ldots, n,\ j = 1, \ldots, n, \\
& y_i \in \{0,1\}, \qquad\quad i = 1, \ldots, n.
\end{aligned} \tag{31}$$

1 番目の制約式は，箱 i が使用されている場合は詰込まれた荷物の重さ合計が上限内に収まることを，箱 i が使用されていない場合は荷物が詰込めないことを表す．2 番目の制約式は，各荷物 j がちょうど 1 つの箱に詰込まれることを表す．

6.4 離接した制約式

一般に，最適化問題では全ての制約式を同時に満たすことを求められるが，現実問題では m 本の制約式のうちちょうど k 本だけを満たすことを求められる場合も少なくない．これは離接した制約式と呼ばれ，選択や順序付けなどの組合せ的な制約条件を記述する場合に用いられる．

例えば，2 つの制約式 $\sum_{j=1}^{n} a_{1j} x_j \leq b_1$ と $\sum_{j=1}^{n} a_{2j} x_j \leq b_2$ $(0 \leq x_j \leq u_j,\ j = 1, \ldots, n)$ の少なくとも一方が成立するという場合は，各制約式に対応する 2 値変数 y_1, y_2 を導入すれば以下の通りに記述できる．

$$\begin{cases}
\displaystyle\sum_{j=1}^{n} a_{1j} x_j \leq b_1 + M(1 - y_1), \\
\displaystyle\sum_{j=1}^{n} a_{2j} x_j \leq b_2 + M(1 - y_2), \\
y_1 + y_2 = 1, \\
y_1, y_2 \in \{0,1\}.
\end{cases} \tag{32}$$

ここで，M は

$$M \geq \max\left\{ \sum_{j=1}^{n} a_{1j}x_j - b_1, \sum_{j=1}^{n} a_{2j}x_j - b_2 \;\middle|\; 0 \leq x_j \leq u_j, j = 1, \ldots, n \right\} \tag{33}$$

を満たす十分に大きな定数（big-M と呼ばれる）である．$y_i = 0$ の場合は，制約式の右辺は $b_i + M$ と十分に大きな値を取り，各変数 x_j の取る値に関わらず必ず満たされる．

以下では，離接した制約式を持つ整数計画問題の例として 1 機械スケジューリング問題と長方形詰込み問題を紹介する．

1 機械スケジューリング問題：n 個の仕事とこれらを処理する 1 台の機械が与えられる．機械は 2 つ以上の仕事を同時には処理できないものとする．仕事 i の処理にかかる時間を $p_i(> 0)$，納期を $d_i(\geq 0)$ とする．s_i と x_{ij} は変数で，s_i は仕事 i の開始時刻，x_{ij} は仕事 i が仕事 j に先行するならば $x_{ij} = 1$，そうでなければ $x_{ij} = 0$ の値を取る．このとき，仕事の納期遅れの合計が最小となる処理スケジュールを求める問題は以下の通りに定式化できる．

$$\begin{array}{lll} \text{minimize} & \displaystyle\sum_{i=1}^{n} \max\{s_i + p_i - d_i, 0\} & \\ \text{subject to} & s_i + p_i \leq s_j + M(1 - x_{ij}), & i = 1, \ldots, n,\ j \neq i, \\ & x_{ij} + x_{ji} = 1, & i = 1, \ldots, n,\ j \neq i, \\ & x_{ij} \in \{0, 1\}, & i = 1, \ldots, n,\ j \neq i, \\ & s_i \geq 0, & i = 1, \ldots, n. \end{array} \tag{34}$$

納期遅れは仕事 i の終了時刻 $s_i + p_i$ が納期 d_i より後になる場合のみ生じるので，各仕事 i に対する納期遅れは $\max\{s_i + p_i - d_i, 0\}$ と記述できる．1 番目の制約式は，仕事 i が仕事 j に先行するならば仕事 i の終了時刻が仕事 j が開始時刻の前になることを表す．2 番目の制約式は，仕事 i が仕事 j に先行するかもしくはその逆が必ず成り立つことを表す．目的関数が最大値の最小化なので，納期遅れを表す新たな変数 $t_i = \max\{s_i + p_i - d_i, 0\}$ を導入すると整数計画問題に変換できる．

$$\begin{array}{lll} \text{minimize} & \displaystyle\sum_{i=1}^{n} t_i & \\ \text{subject to} & s_i + p_i \leq s_j + M(1 - x_{ij}), & i = 1, \ldots, n,\ j \neq i, \\ & x_{ij} + x_{ji} = 1, & i = 1, \ldots, n,\ j \neq i, \\ & s_i + p_i - d_i \leq t_i, & i = 1, \ldots, n, \\ & x_{ij} \in \{0, 1\}, & i = 1, \ldots, n,\ j \neq i, \\ & s_i, t_i \geq 0, & i = 1, \ldots, n. \end{array} \tag{35}$$

長方形詰込み問題：図 7 に示すように，幅が固定で十分な高さがある長方形の容器と n 個の長方形の荷物が与えられる．容器の幅を W，各荷物 i の幅を w_i，高さを h_i とする．荷物はその下辺が容器の下辺と平行になるように配置し，回転は許さないものとする．ここで，全ての荷物を互いに重ならないように容器内に配置する．(x_i, y_i) を荷物 i の左下隅の座標を表す変数とすると（容器の左下隅を原点とする），問題の制約条件は以下の通りに記述できる．

制約条件 1：　荷物 i は容器内に配置される．

これは，以下の 2 本の不等式がともに成り立つことと同値である．

$$\begin{aligned} 0 \leq x_i \leq W - w_i, \\ 0 \leq y_i \leq H - h_i. \end{aligned} \tag{36}$$

制約条件 2：　荷物 i, j は互いに重ならない．

これは，以下の 4 本の不等式のうち 1 本以上が成り立つことと同値であり，各不等式はそれぞれ荷物 i が荷物 j の左側，右側，下側，上側にあることを記述している．

$$\begin{aligned} x_i + w_i \leq x_j, \\ x_j + w_j \leq x_i, \\ y_i + h_i \leq y_j, \\ y_j + h_j \leq y_i. \end{aligned} \tag{37}$$

$z_{ij}^{\mathrm{left}}, z_{ij}^{\mathrm{right}}, z_{ij}^{\mathrm{lower}}, z_{ij}^{\mathrm{upper}}$ は変数で，それぞれ荷物 i が荷物 j の左側，右側，下側，上側にあるならば 1，そうでなければ 0 の値を取る．このとき，制約条件を満たした上で必要な容器の高さ H を最小にする荷物の配置を求める問題は以下の通りに定式化できる．

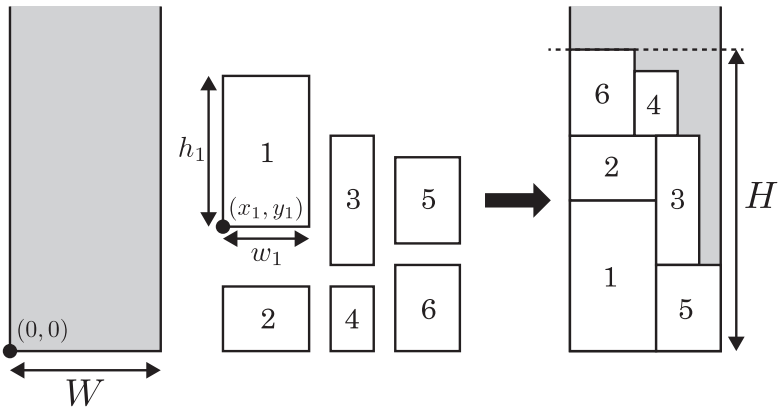

図 7　長方形詰込み問題の例

$$
\begin{array}{lll}
\text{minimize} & H & \\
\text{subject to} & 0 \le x_i \le W - w_i, & i = 1, \ldots, n, \\
& 0 \le y_i \le H - h_i, & i = 1, \ldots, n, \\
& x_i + w_i \le x_j + M(1 - z_{ij}^{\text{left}}), & i = 1, \ldots, n, j \ne i, \\
& x_j + w_j \le x_i + M(1 - z_{ij}^{\text{right}}), & i = 1, \ldots, n, j \ne i, \\
& y_i + h_i \le y_j + M(1 - z_{ij}^{\text{lower}}), & i = 1, \ldots, n, j \ne i, \\
& y_j + h_j \le y_i + M(1 - z_{ij}^{\text{upper}}), & i = 1, \ldots, n, j \ne i, \\
& z_{ij}^{\text{left}} + z_{ij}^{\text{right}} + z_{ij}^{\text{lower}} + z_{ij}^{\text{upper}} = 1, & i = 1, \ldots, n, j \ne i, \\
& z_{ij}^{\text{left}}, z_{ij}^{\text{right}}, z_{ij}^{\text{lower}}, z_{ij}^{\text{upper}} \in \{0, 1\}, & i = 1, \ldots, n, j \ne i.
\end{array} \tag{38}
$$

6.5　非線形関数

非凸関数最小化問題は整数計画問題に近似できる．図 8 に示すように，非凸関数 $f(x)$ 上の m 個の点 $(a_1, f(a_1)), \ldots, (a_m, f(a_m))$ を適当に選んで線分で繋ぐと区分線形関数 $g(x)$ が得られる．区分線形関数上の点 $(x, g(x))$ はある線分上にある．例えば，点 $(x, g(x))$ が $(a_i, f(a_i))$ と $(a_{i+1}, f(a_{i+1}))$ で結ばれる線分上にある場合は以下の通りに記述できる．

$$
\begin{cases}
(x, g(x)) = t_i(a_i, f(a_i)) + t_{i+1}(a_{i+1}, f(a_{i+1})), \\
t_i + t_{i+1} = 1, \\
t_i, t_{i+1} \ge 0.
\end{cases} \tag{39}
$$

これを考慮すると一般の場合も以下の通りに記述できる．

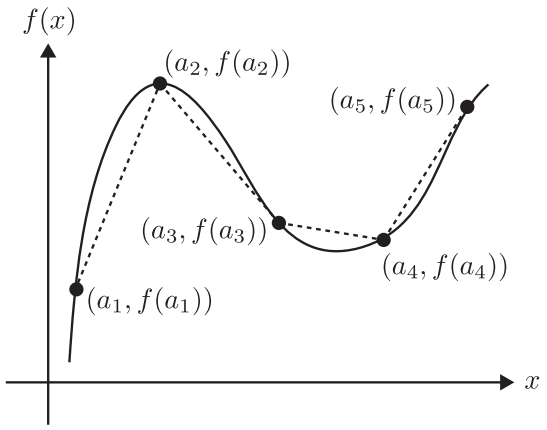

図 8　非凸関数の区分線形関数による近似

$$\begin{cases} (x, g(x)) = \sum_{i=1}^{m} t_i (a_i, f(a_i)), \\ \sum_{i=1}^{m} t_i = 1, \\ t_i \geq 0, \qquad\qquad i = 1, \ldots, m, \\ \text{高々 2 つの隣り合う } t_i \text{ が正.} \end{cases} \tag{40}$$

ここで，2 値変数 $z_1, \ldots, z_{m-1}$ を導入すると「高々 2 つの隣り合う t_i が正」という制約条件は以下の通りに記述できる．

$$\begin{cases} t_1 \leq z_1, \\ t_i \leq z_{i-1} + z_i, \quad i = 2, \ldots, m-1, \\ t_m \leq z_{m-1}, \\ \sum_{i=1}^{m-1} z_i = 1, \\ z_i \in \{0, 1\}, \qquad i = 1, \ldots, m-1. \end{cases} \tag{41}$$

次に，2 値変数で定義される非線形関数を線形関数に変換する方法を紹介する．まず，2 値変数 x_1 と x_2 の積 $y = x_1 x_2$ を考える．このとき，(x_1, x_2, y) の実行可能解は $(0,0,0), (1,0,0), (0,1,0), (1,1,1)$ の 4 通りなので以下の通りに記述できる．

$$\begin{cases} y \geq x_1 + x_2 - 1, \\ y \leq x_1, \\ y \leq x_2, \\ x_1, x_2 \in \{0, 1\}. \end{cases} \tag{42}$$

これらの制約式は実行可能解全体の凸包から得られる．同様に k 個の 2 値変数の積 $y = \prod_{i=1}^{k} x_i$ も以下の通りに記述できる．

$$\begin{cases} y \geq \sum_{i=1}^{k} x_i - (k-1), \\ y \leq x_i, \qquad\qquad i = 1, \ldots, k, \\ x_i \in \{0, 1\}, \qquad\qquad i = 1, \ldots, k. \end{cases} \tag{43}$$

6.6 グラフの連結性

グラフにおける最適化問題では選択した部分グラフの連結性が求められる場合が少なくない．ここでは，グラフの連結性を制約条件に持つ整数計画問題の例として最小全域木問題と巡回セールスマン問題を紹介する．

無向グラフ $G=(V,E)$ の任意の頂点 $i,j\in V$ の間に路が存在するならば G は連結であると呼ぶ．図 9 は連結なグラフと非連結なグラフの例である．これは，任意の頂点集合 $S\subset V$ $(S\neq\emptyset)$ に対して，S と $V\setminus S$ の間を繋ぐ辺が少なくとも 1 本は存在するという制約条件に置き換えられる．

最小全域木問題：無向グラフ $G=(V,E)$ と各辺 $(i,j)\in E$ の長さ d_{ij} が与えられる．閉路を持たない連結な部分グラフは木，全ての頂点を繋ぐ木は全域木と呼ばれる．x_{ij} は変数であり，辺 (i,j) は木に含まれるならば $x_{ij}=1$，そうでなければ $x_{ij}=0$ の値を取る．このとき，辺の長さの合計が最小となる全域木を求める問題は以下の通り定式化できる．

$$\begin{aligned}
\text{minimize}\quad & \sum_{(i,j)\in E} d_{ij}x_{ij} \\
\text{subject to}\quad & \sum_{i\in S}\sum_{j\in V\setminus S} x_{ij}\geq 1, \quad S\subset V,\ S\neq\emptyset \\
& \sum_{(i,j)\in E} x_{ij}=n-1, \\
& x_{ij}\in\{0,1\}, \qquad (i,j)\in E.
\end{aligned} \tag{44}$$

1 番目の制約式は，辺集合 $T\subseteq E$ が全ての頂点を連結することを表し，カットセット制約と呼ばれる．2 番目の制約式は，$|T|=n-1$ を満たすことを表す．これらの制約式は T が全域木となるための必要十分条件である．

巡回セールスマン問題：無向グラフ $G=(V,E)$ の全ての頂点をちょうど 1 回ずつ通る閉路は巡回路と呼ばれる．巡回路となるためには，各頂点 k に接続する辺がちょうど 2 本でなければならない．しかし，これだけでは不十分で，図 10（左）に示すような部分巡回路を排除する必要がある．これは，任意の頂点集合 $S\subset V$ $(S\neq\emptyset)$ に含まれる辺の本数が $|S|-1$ 以下であるという制約条件に置き換えられる．

無向グラフ $G=(V,E)$ と各辺 $(i,j)\in E$ の長さ d_{ij} が与えられる．x_{ij} は変数で，辺 $(i,j)\in E$ が巡回路に含まれるならば $x_{ij}=1$，そうでなければ $x_{ij}=0$ の値を取る．このとき，全ての頂点をちょうど 1 回ずつ訪問する最短の巡回路を求める問題は以下の通りに定式化できる．

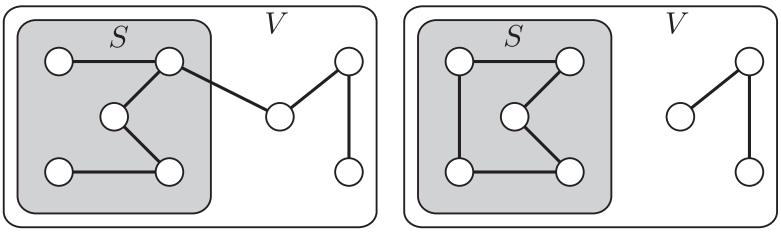

図 9　連結なグラフ（左）と非連結なグラフ（右）の例

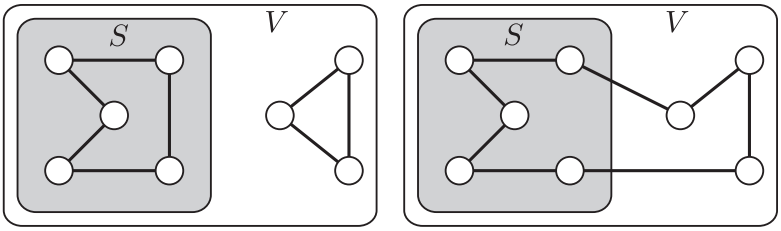

図 10　部分巡回路（左）と巡回路（右）の例

$$
\begin{aligned}
\text{minimize} \quad & \sum_{(i,j)\in E} w_{ij}x_{ij} \\
\text{subject to} \quad & \sum_{(i,k)\in E:i<k} x_{ik} + \sum_{(k,j)\in E:k<j} x_{kj} = 2, \quad k \in V, \\
& \sum_{(i,j)\in E:i,j\in S} x_{ij} \leq |S| - 1, \qquad S \subset V,\ S \neq \emptyset, \\
& x_{ij} \in \{0,1\}, \qquad (i,j) \in E.
\end{aligned}
\tag{45}
$$

1 番目の制約式は，各頂点に接続する辺がちょうど 2 本となることを表す．2 番目の制約式は，部分巡回路を持たないことを表し，部分巡回路除去制約と呼ばれる．

最小全域木問題のカットセット制約や巡回セールスマン問題の部分巡回路除去制約は，制約式の数が $\mathrm{O}(2^n)$ と膨大で，全ての制約式を書き下して整数計画ソルバーに解かせるのは現実的ではないため，必要に応じて制約式を逐次追加する切除平面法が必要となる．

グラフの連結性を制約条件に持つ整数計画問題の定式化と解法については (藤江 2011; 久保，ペドロソ，村松，レイス 2012) が詳しく，新たな変数を導入して必要な制約式の数を抑える方法が紹介されている．

7　最適解が求められない場合の対処法

最近の整数計画ソルバーは非常に高性能ではあるものの，解候補を体系的に列挙する分枝限定法を探索の基本戦略とするため，与えられた問題例によってはいつまで待っても計算が終了しない場合が少なくない．ここでは，目的関数の値を最小化する整数計画問題を考える．分枝限定法は，整数計画問題を分枝操作によって小規模な部分問題に分解しつつ，各部分問題では，暫定解から得られる最適値の上界値と，線形計画緩和問題から得られる最適値の下界値を利用した限定操作によって無駄な探索を省いている．そのため，いつまで待っても整数計画ソルバーの計算が終了しないならば，(1) 線形計画緩和問題の求解に多大な計算時間を要する，(2) 限定操作が効果的に働いていないことなどが原因として考えられる．もちろん，整数計画ソルバー

は分枝限定法以外にも多くのアルゴリズムを内包しているため，これだけが原因であると決めつけるべきではないが，対策を練る上でまず始めに確認すべき事項である．

(1) については，原問題から各変数の整数条件を取り除いた線形計画問題を整数計画ソルバーで解けば計算時間を見積もることができる．実際には，整数計画ソルバーは再最適化と呼ばれる手法を利用するため，整数計画問題の各部分問題において線形計画緩和問題の求解に要する計算時間はもっと短くなる．しかし，この方法で線形計画問題を 1 回解くのに要する計算時間が長いと感じるようであれば，問題例の規模が整数計画ソルバーで解くには大き過ぎると判断するのが妥当であろう．ただし，集合被覆問題や集合分割問題などの線形計画緩和問題では，単体法と内点法で計算時間が大きく異なるため，（部分問題ではなく）原問題の線形計画緩和問題に適用するアルゴリズムを切り替えることで計算時間を大幅に削減できる場合もある．

(2) については，(i) 暫定解から得られる最適値の上界値が悪い場合，(ii) 線形計画緩和問題から得られる最適値の下界値が悪い場合，(iii) 多数の最適解が存在する場合などが考えられる．これらは，整数計画ソルバーの実行時に出力される最適値の上界値と下界値から確認できる．

まず，(i) 暫定解から得られる上界値が悪い場合を考える．これは，実行可能解が非常に少ないかもしくは存在しないため，整数計画ソルバーの実行時に良い実行可能解を発見できないことが原因として考えられる．このような場合は，制約式を必ず満たさなければならない制約式（絶対制約）とできれば満たして欲しい制約式（考慮制約）に分けた上で，優先度の低い考慮制約を緩和する方法がある．例えば，制約式 $\sum_{j=1}^{n} a_{ij}x_j \geq b_i$ を $\sum_{j=1}^{n} a_{ij}x_j \geq b_i - \varepsilon$（$\varepsilon$ は適当な正の定数）に置き換える方法や，新しい変数 $z_i(\geq 0)$ とペナルティ係数 $p_i(> 0)$ を導入して $\sum_{j=1}^{n} a_{ij}x_j \geq b_i - z_i$ に置き換えた上で目的関数に新たな項 $+p_iz_i$ を加える方法などがある．また，利用者の持つ先験的な知識を利用して容易に実行可能解を求められるならば，利用者が持つアルゴリズムで求めた実行可能解を初期暫定解として整数計画ソルバーに与えることも可能である．

次に，(ii) 線形計画緩和問題から得られる最適値の下界値が悪い場合を考える．図 11 に示すように，線形計画緩和問題の実行可能領域は，整数計画問題の実行可能解となる整数格子点のみを含む凸多面体となるため，同じ整数計画問題に対して線形計画緩和問題の最適値が異なる複数の定式化が存在する．つまり，整数計画問題では最適値の良い下界値が得られる強い定式化と，そうでない弱い定式化が存在する．ちなみに，最も強い定式化は整数計画問題の実行可能解全体の凸包を記述することであるが，凸包を記述する全ての制約式を求めることは，最悪の場合には全ての実行可能解を列挙することに他ならないため現実的な方法ではない．たしかに，制約式の数が少ない定式化の方が見栄えも良く，分枝限定法を適用した際にも各部分問題における線形計画緩和問題の求解に要する計算時間も短くなるように思われる．しかし，最適値の上界値と下界値の差が広がれば分枝限定法で生成される部分問題の数は急激に増加するため，安易に制約式を減らすべきではない．一方で，多くの整数計画ソルバーは冗長な制約式を

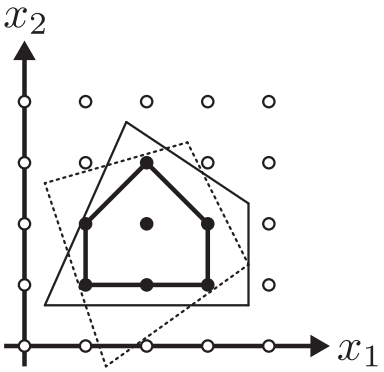

図 11　整数計画問題の実行可能解を含む凸多面体の例

前処理で除去するため，制約式が多少増えても計算時間にはあまり影響しない場合が多い．例えば，与えられた現実問題を完全単摸行列に近い形の制約行列を持つ整数計画問題に定式化できる場合は，線形計画緩和問題から良い下界値が得られることが期待できる．

最後に，(iii) 多数の最適解を持つ場合を考える．最適値の上界値と下界値の差が小さいにも関わらず，いつまで待っても整数計画ソルバーの計算が終了しないならば，整数計画問題が多数の最適解を持っている可能性がある．このような場合は，目的関数や制約式を変更して最適解の数を減らす方法がある．例えば，6.3 節で紹介したビンパッキング問題の定式化では，使用する箱の数が最小であれば使用する箱の組合せは何でも構わないため多数の最適解が生じる．そこで，必ず番号の小さい箱から順に使用するという制約式を追加すると最適解の数を減らすことができる．

$$y_i \geq y_{i+1}, \quad i = 1, \ldots, n-1. \tag{46}$$

また，5.2 節で紹介した多目的最適化問題の定式化では，1 変数からなる目的関数を持つ整数計画問題に変換するとやはり多数の最適解が生じる．このような場合は，いつまで待っても整数計画ソルバーの計算が終了しないならば線形和を最小化する定式化に変更した方が良い．また，目的関数 $\sum_{j=1}^{n} c_j x_j$ の各項の係数 c_j が全て同じ値を取る場合も多数の最適解が生じ易いため，可能ならば各項の係数 c_j をいろいろな値に変えて最適解の数を絞り込む方が良い．

最後に，いつまで待っても整数計画ソルバーの計算が終了しない場合には，最適解を求めることを諦めるのも 1 つの手である．整数計画ソルバーは探索中に得られた暫定解を保持しているので，与えられた計算時間内に最適解が求められなくても良い実行可能解が求まれば，利用者によっては十分に満足できる場合も多い．また，整数計画ソルバーは線形計画緩和問題を解いて得られる最適値の下界値も保持しているので，事後にはなるが得られた暫定解の精度も評価できる．実際に，整数計画ソルバーは近似解法としても高性能であり，メタヒューリスティク

スなどの発見的解法を利用もしくは開発する前に，整数計画ソルバーで良い実行可能解が得られるかどうか確認するべきである．最適解が求められない場合の対処法については (宮代 2014; 宮代，松井 2006) が詳しい．

8 おわりに

本論文では，数理最適化の専門家ではない利用者が，現実問題の解決に取り組む際に必要となる整数計画ソルバーの基本的な利用法と定式化の技法を解説した．整数計画ソルバーの解説は本論文が初めてではなく，オペレーションズ・リサーチの分野では同じ趣旨の解説がいくつか発表されている (藤江 2012; 宮代，松井 2006; 宮代 2012)．また，現実問題を線形計画問題や整数計画問題に定式化する技法については (久保 他 2012; Williams 2013) が詳しい．これらの文献は，本論文では取り上げていない多くの内容を含んでいるので，整数計画ソルバーに興味を持たれた読者にはぜひ一読をお勧めする．

参考文献

Achterberg, T. and Wunderling, R. (2013). “Mixed Integer Programming: Analyzing 12 Years of Progress.” In Jünger, M. and Reinelt, G. (Eds.), *Facets of Combinatorial Optimization—Festschrift for Martin Grötschel*, pp. 449–481. Springer.

Ashford, R. (2007). “Mixed Integer Programming: A Historical Perspective with Xpress-MP.” *Annals of Operations Research*, **149**, pp. 5–17.

Atamtürk, A. and Savelsbergh, M. W. P. (2005). “Integer-programming software systems.” *Annals of Operations Research*, **140**, pp. 67–124.

Berthold, T., Gleixner, A. M., Heinz, S., Koch, T., and Shinano, Y. (2012). SCIP Optimization Suite を利用した混合整数（線形／非線形）計画問題の解法. テクニカル・レポート, Zuse Institute Berlin.

Bixby, R. and Rothberg, E. (2007). “Progress in Computational Mixed Integer Programming—A Look Back from the Other Side of the Tipping Point.” *Annals of Operations Research*, **149**, pp. 37–41.

Charnes, A. and Cooper, W. W. (1961). *Management Models and Industrial Applications of Linear Programming*. John Wiley & Sons.

Chvatal, V. (1983). *Linear Programming*. W.H.Freeman & Company.

DeNero, J. and Klein, D. (2008). “The Complexity of Phrase Alignment Problems.” In *Proceedings of 46th Annual Meeting of the Association for Computational Linguistics: Human*

Language Technology, Short Papers, pp. 25–28.

Filatova, E. and Hatzivassiloglou, V. (2004). "A Formal Model for Information Selection in Multi-Sentence Text Extraction." In *Proceedings of Coling 2004*, pp. 397–403.

Fourer, R. (2013). "Linear programming." *OR/MS Today*, **40** (3), pp. 40–53.

藤江哲也 (2011). 最近の混合整数計画ソルバーの進展について. オペレーションズ・リサーチ, **56** (5), pp. 263–268.

藤江哲也 (2012). 整数計画法による定式化入門. オペレーションズ・リサーチ, **57** (4), pp. 190–197.

Gillick, D. and Favre, B. (2009). "A Scalable Global Model for Summarization." In *Proceedings of the Workshop on Integer Programming for Natural Language Processing at the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 10–18.

平尾努, 鈴木潤, 磯崎秀樹 (2009). 最適化問題としての文書要約. 人工知能学会論文誌, **24** (2), pp. 223–231.

Kellerer, H., Pferschy, U., and Pisinger, D. (2004). *Knapsack Problems*. Springer.

Koch, T., Achterberg, T., Andersen, E., Bastert, O., Berthold, T., Bixby, R. E., Danna, E., Gamrath, G., Gleixner, A. M., Heinz, S., Lodi, A., Mittelmann, H., Ralphs, T., Salvagnin, D., Steffy, D. E., and Wolter, K. (2011). "MIPLIB2010: Mixed Integer Programming Library version 5." *Mathematical Programming Computation*, **3** (2), pp. 103–163.

今野浩 (1982). 整数計画法と組合せ最適化. 日科技連.

今野浩 (1987). 線形計画法. 日科技連.

今野浩 (2005). 役に立つ 1 次式—整数計画法「気まぐれな王女」の 50 年. 日本評論社.

今野浩 (2014). ヒラノ教授の線形計画法物語. 岩波書店.

Korte, B. and Vygen, J. (2012). *Combinatorial Optimization: Theory and Algorithms (5th edition)*. Springer.

越川満, 内山将夫, 梅谷俊治, 松井知己, 山本幹雄 (2010). 統計的機械翻訳におけるフレーズ対応最適化を利用した N-best 翻訳候補のリランキング. 情報処理学会論文誌, **51** (8).

久保幹雄, J. P. ペドロソ, 村松正和, A. レイス (2012). あたらしい数理最適化 — Python 言語と Gurobi で解く. 近代科学社.

Linderoth, J. T. and Ralph, T. K. (2006). "Noncommercial Software for Mixed-Integer Linear Programming." In Karlof, J. K. (Ed.), *Integer Programming—Theory and Practice*, pp. 253–303. Taylor & Francis.

Lodi, A. (2010). "Mixed Integer Programming Computation." In Jünger, M., Liebling, T., Naddef, D., Nemhauser, G., Pulleyblank, W., Reinelt, G., Rinaldi, G., and Wolsey, L. (Eds.), *50 Years of Integer Programming 1958-2008*, pp. 619–645. Springer.

Ma, X. (2006). “Champollion: A Robust Parallel Text Sentence Aligner.” In *Proceedings of 5th International Conference on Language Resources and Evaluation*, pp. 489–492.

McDonald, R. (2007). “A Study of Global Inference Algorithms in Multi-Document Summarization.” In *Proceedings of the European Conference on Information Retrieval (ECIR)*, pp. 557–564.

Mittelmann, H. D. (2014). “Decision Tree for Optimization Software.” `http://plato.asu.edu/guide.html`.

宮代隆平 (2012). 整数計画ソルバー入門. オペレーションズ・リサーチ, **57** (4), pp. 183–189.

宮代隆平 (2014). 整数計画法メモ. `http://www.tuat.ac.jp/~miya/ipmemo.html`.

宮代隆平, 松井知己 (2006). ここまで解ける整数計画. システム／制御／情報, **50** (9), pp. 363–368.

Moore, R. C. (2002). “Fast and Accurate Sentence Alignment of Bilingual Corpora.” In *Machine Translation: From Research to Real Users*, pp. 135–144. Springer.

Nemhauser, G. L. and Wolsey, L. A. (1988). *Integer and Combinatorial Optimization*. John Wiley & Sons.

西川仁，平尾努，牧野俊朗，松尾義博，松本裕治 (2013). 冗長性制約付きナップサック問題に基づく複数文書要約モデル. 自然言語処理, **20** (4), pp. 586–612.

西野正彬，平尾努，永田昌明 (2013). 集合パッキング問題に基づく文アラインメントのモデル化. 言語処理学会第 19 回年次大会発表論文集, pp. 932–935.

西野正彬，鈴木潤，梅谷俊治，平尾努，永田昌明 (2014). 列生成法を用いた高速なアラインメント. 言語処理学会第 20 回年次大会発表論文集, pp. 364–367.

Schrijver, A. (1998). *Theory of Linear and Integer Programming*. John Wiley & Sons.

高村大也，奥村学 (2008). 最大被覆問題とその変種による文書要約モデル. 人工知能学会論文誌, **23** (6), pp. 505–513.

高村大也，奥村学 (2010). 施設配置問題による文書要約のモデル化. 人工知能学会論文誌, **25** (1), pp. 174–182.

Williams, H. P. (2013). *Model Building in Mathematical Programming (5th edition)*. John Wiley & Sons.

Wolsey, L. A. (1998). *Integer Programming*. John Wiley & Sons.

略歴

梅谷俊治：1998 年大阪大学大学院基礎工学研究科博士前期課程修了．2002 年京都大学大学院情報学研究科博士後期課程指導認定退学．豊田工業大学助手，電気通信大学助教を経て，2008 年より大阪大学大学院情報科学研究科准教授，現在に至る．博士（情報学）．専門は組合せ最適化，アルゴリズム．日本オ

ペレーションズ・リサーチ学会，情報処理学会，人工知能学会，INFORMS，
MOS 各会員．

（2014 年 5 月 22 日　受付）
（2014 年 6 月 13 日　採録）